Mem. Fac. Eng., Osaka City Univ., Vol. 43, pp.43−58 (2002)

Ad-Hoc Network Simulator Based on DSDV Routing Method

Hisayoshi SUGIYAMA*, Tetsuo TSUJIOKA** and Masashi MURATA***

(Received September 30, 2002)

Synopsis

A simulator is introduced by which the operation of Ad-Hoc network is simulated on display screen of UNIX workstation. The routing protocol is based on Destination Sequence Distance Vector (DSDV). Number of nodes, communication range, broadcast frequency of forwarding table, and other parameters inherent in the DSDV protocol can be fixed arbitrary by 'Initialization File'. As the simulation step proceeds, noticiable events raised in the network operation are indicated graphically in the simulation window. Forwarding tables of each node are also indicated by other windows. Almost of all parameters can be changed arbitrary when the simulation pauses at the appointed time. Finally, the simulation results of each node including transmissin collision time and wait time for avoiding that are recorded in data files.

KEYWORDS: Ad-Hoc, DSDV, simulation, simulator, UNIX

1. Introduction

Ad-Hoc network has been studied as the wireless network with no center station consists of mobile nodes configure a group incidentally[1]. In this mobile network, common radio channel is provided and through this channel, communication link is established between terminals so close to each other that they can transmit signals immediately. Any terminals in the network can transmit packets to each other through the communication path which traces the series of communication links established by intermediate terminals. Ad-Hoc network is expected to used in the incidental network such as that of laptop computers used by members in conference, or more urgent case of communications in disaster area.

This paper introduces a simulater by which the operation of Ad-Hoc network is simulated on display screen of UNIX workstation. Its routing protocol is based on Destination Sequence Distance Vector (DSDV)[2]. Parameters including number of nodes, communication range, broadcast frequency of forwarding table inherent in the DSDV protocol can be fixed arbitrary by 'Initialization File'. Noticiable events raised in the network operation are indicated graphically in the simulation window. Forwarding tables of each node are also indicated by other windows. Almost of all parameters can be changed arbitrary when the simulation pauses at the appointed time. Simulation data are dumped to text files including transmissin collision time and wait time for avoiding that at each node when the simulation is terminated.

In Section 2, Ad-Hoc network and DSDV routing method are explained. Firstly, forwarding table which each node uses to determine the communication pass to some destination, and the routing method following DSDV and forwarding table are described. Secondly, the process of broadcasting update from each node is described. This broadcast of update maintains the network even when each node moves and changed its relative position to other nodes. The details of entries recorded in the update and the process of the network maintenance are exlained. Thirdly, one of the serious problem *hidden terminal problem* in Ad-Hoc network is described with signal collisions occurred in the network. Finally, details of network re-construction inherent in the Ad-Hoc network is explained. This re-construction is done in severel processes following the DSDV method and the example is shown in one of the process of the re-construction.

---

* Associate Professor, Department of Physical Electronics and Informatics
** Research Associate, Department of Physical Electronics and Informatics
*** Professor, Department of Physical Electronics and Informatics

In Section 3, details of DSDV simulator including how to download it, how to compile it, and how to execte it. Firstly, the overview of the simulator is descrived along with the flow chart which represents the process of execution. Secondly, download site of this simulator is indicated and details of uncompression, compilation, and execution are described. Thirdly, the details are described of initialization file which specifies the initial parameters for the simulation. Fourthly, details of the information indication during the simulation is described including the state of each node and that of network. Finally, details of the proceeding of the simulation is described including the restriction of indication and of X server operation for the rapid iteration during the simulation.

2. Fundamentals

Ad-Hoc network is a kind of mobile network as a set of mobile terminals communicating each other through pre-assinged radio channel. In Ad-Hoc network, radio communication link is established between terminals so close to each other that they can transmit signals immediately. Any terminals in the network can transmit packets to each other through the communication path which traces the series of communication links established by intermediate terminals. In this Section, the principles of Ad-Hoc network and DSDV which this simulator adapts as the routing protocol of the network are explained.

2.1 Ad-Hoc network and DSDV protocol

Ad-Hoc network can be classified into two classes according to the method by which each terminals determines the communication pass to the destination terminal (from now on, simply referred to as the *destination*): *Table driven method* and *On demand method*.

According to the former (Table driven) method, each terminal keeps the routing table in which the pairs of destination and *next hop* are recorded. Here, the next hop means that the terminal among ones with which communication link is established and that the packet is transmitted to the terminal as the first relay terminal to the destination. Because the network topology may be modified with the movements of terminals, the routing table should be updated periodically.

According to the latter (On demand) method[3], each terminal searchs the communication pass to the destination anytime when the terminal intends to transmit packets to the destination. This search begins when the terminal broadcasts inquiry packet and finishes when the destination receives the packet and returns it to the original terminal with the intermediate terminals recorded in the packet.

DSDV belongs to the Table driven method above descrived. DSDV is the abbreviation of *Destination Sequence Distance Vector*. This can be distinguished from other methods in the same class, by the time stamp of source terminal added to routing table update information (from now on, this information is simply referred to as *update*). The time stamp is called *sequence* and this sequence avoids the closed loop appears in communication pass.

In DSDV, the routing table kept by each terminal is called *forwarding table* (from now on, forwarding table is denoted by 'F. T.'). Each terminal broadcasts update periodically and the terminal which receives this information updates its F.T.. The broadcast follows the *flooding* method by which each terminal transmits the information to all the terminals with which the communication link is established. This flooding method disperses the update all over the network immediately. On the other hand, the communication pass to the destination is determined according to the F.T. when a terminal intends to transmit a packet. In the following subsectiones, these subjects constituting DSDV protocol are explained, and example of network re-construction is shown when network topology is modified along with the movement of terminals.

2.2 Forwarding table and routing

An example of Ad-Hoc network and that of F.T. based on DSDV are shown in Fig. 1. Four nodes (node equals terminal in actual network) construct the network in this example. Restricted communication range is defined for each node and pair of nodes within this ranges of each other establish the communication link. In Fig. 1, the pair of node A and B, that of node B and C, and that of node B and D establish the communication link. Two nodes, which locate without communication ranges of each other, communicate through communication pass which traces a series of communication links. This communication pass is determined according to the F.T. of each node.

Examples of F.T. according to the exemplified network are indicated in Fig. 1. In each F.T., the following six terms are recorded for one destination (the set of these terms are called *entry*).

dst represents the destination.

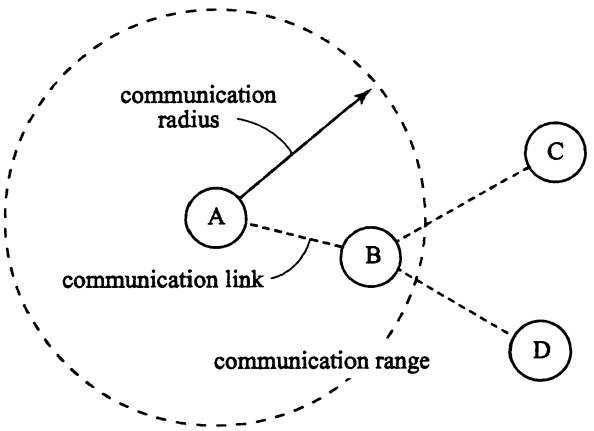
nxt represents the next hop node.

mtr represents the metric to the destination.

seq represents the sequence recorded by the node which generates this entry.

ist represents the installed (or updated) time of this entry.

flg represents the flag which indicates whether or not this entry is transmitted at the next broadcast.



F. T. of node A

dst	nxt	mtr	seq	ist	flg
A	A	0	86	0	*
B	B	1	84	0	*
C	B	2	76	−3	−
D	B	2	86	−1	−

F. T. of node B

dst	nxt	mtr	seq	ist	flg
B	B	0	86	0	*
A	A	1	80	−3	−
C	C	1	76	−4	−
D	D	1	86	−3	−

F. T. of node C

dst	nxt	mtr	seq	ist	flg
C	C	0	86	0	*
A	B	2	80	−1	−
B	B	1	82	0	*
D	B	2	86	−1	−

F. T. of node D

dst	nxt	mtr	seq	ist	flg
D	D	0	94	0	*
A	B	2	80	−2	−
B	B	1	84	0	−
C	B	2	34	−26	−

Fig. 1 Ad Hoc network and forwarding tables of each node depending on DSDV routing method.

Among these terms, the combination of 'dst' and 'nxt' are used to determine the communication pass. 'mtr' and 'seq' are used to determine whether or not each entry is overwrited when new update is received by broadcast. 'ist' means the oldness of the entry in comparison with current time. If 'ist' of the entry exdeeds the predefined value, and if the destination of which establishes communication link with the node, the node recognizes that the communication link is broken and broadcast this information. On the other hand, the node simply erases the entry when 'ist' of other type of entry exdeeds the (another) predefined value. (The details of update process, of broadcast information of broken link, and of elimination of entry are described later.)

Each node always contains one entry which addresses itself (the destination of which is the node itself). This entry is called *basic entry*. The node can increase the 'seq' of this basic entry only. This increase occurs when each node broadcast the update and then 'seq' of the basic entry is added two for being kept even

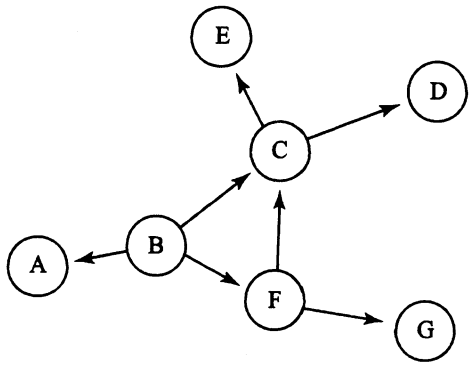
number. There is an exception case in which the node adds one to 'seq' of other entry but basic one, then 'seq' changes to odd number. This case occurs when the node recognizes the broken of communication link. On the other hand, 'ist' is decreased one every time when the node broadcast update, and is reset when it is overwrited by update.

The process of determination of communication pass is exemplified as follows, according to the F.T. of node A in Fig. 1. Now, node A is assumed to intend to transmit packet addressed to node D. Firstly, A recognizes that the next hop node is B by the fourth entry of its F.T.. Therefore, A transmits this packet to B. When node B receives this packet, it recognizes the next hop node corresponds with the destination by the fourth entry of its F.T.. The B transmits the packet to D. Following this process, the communication pass from node A to D is determined as A $\rightarrow$ B $\rightarrow$ D.

As is described above, communication pass in the network is determined according to F.T. in DSDV. Firstly, each node has only basic entry in its F.T., and it takes other entries addressed to other nodes one by one when the node receives updates. Finally, F.T. is completed by entries addressed to all of other nodes in the network. The principle of this process is described in the next subsection.

2.3 Broadcast of update information

Each node broadcasts update periodically in Ad-Hoc network based on DSDV method. The example of this process is shown in Fig. 2. This figure shows that node B broadcasts update in the network with seven nodes. It is assumed that the network is in the initial state and that F.T. of each node contains only basic entry.



F. T. of node B

dst	nxt	mtr	seq	ist	flg
B	B	0	0	0	*

F. T. of node A

dst	nxt	mtr	seq	ist	flg
A	A	0	0	0	*

F. T. of node A

dst	nxt	mtr	seq	ist	flg
A	A	0	0	0	*
B	B	1	0	0	*

Fig. 2 Broadcast for Forwarding Table updating

The update contains the entris which the originating node updated and not broadcasted, and basic entry of the node. The F.T. of node B is depicted at upper right of Fig. 2. This contains only basic entry. Therefore, node B broadcasts only this basic entry as update at the beginning.

Firstly, the update transmitted form node B is received by node A, C, and F within communication range of node B (from now on, the other nodes within communication range of a node are referred to as *adjacent nodes* to the latter node). The node which receives the update compares 'dst', 'mtr', and 'seq' of each entry with each ones recorded in its F.T. (from now on, entries in the received update are referred to as *received entries*, entries recorded in its own F.T. is referred to as *self entry*). If 'dst' of received entry is not found in any of self entries, the received entry is added to F.T.. On the other hand, if 'dst' of received entry is found in one of self entries, only in the case when the following conditions:

1. 'seq' of received entry exceeds that of self entry which addresses the same destination as the received one (i.e., the time when the received entry generated is later than that of corresponding self entry).
2. If 'seq's of both entries are equals to each other, 'mtr' of the received entry is smaller (i.e., the hop

count required to reach the destination is smaller) than that of self entry.

are satisfied, the self entry is overwrited by the received entry. In this case (and the case when received entry is added to F.T.), 'nxt' and 'mtr' recorded in the entry are changed as follows:

1. 'nxt' is changed to the adjacent node which transmits the update.
2. 'mtr' is added one considering the one hop count between the adjacent node which transmits the update.

Therfore, the comparison of 'mtr's above described is actually that of (mtr + 1) of received entry and (mtr) of self entry.

For example, the F.T. of node A contains only basic entry at the initial state, and the F.T. takes new entry addressed to node B when node A receives update form B. This process is indicated in the right part of Fig. 2.

As is described before, each node broadcasts update periodically. Adjacent nodes A, C, and F broadcast update temporally when individually determined next transmission time comes to each node. For example, node C transmits update to node B, D, E, and F. On the other hand, node F transmits update to node B, C, and G. These updates transmitted contain basic entry of node B. Therefore, finally the basic entry of node B (i.e., the communication pass to node B) is included by F.T.s of every node after enough time for broadcasting to each other passed.

This process occurs for basic entries of every other node, and after enough time passed, F.T. of each node is completed by entries addressed to every other node.

Broadcast of update information is necessary for the configuration and maintenance of Ad-Hoc network. When the F.T. is completed, packet transmission to some specified node (single cast) is possible. For example, if node A transmits packet addressed to node D, this packet is relayed to the destination D through communication pass: A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D. This process is shown in Fig. 3.

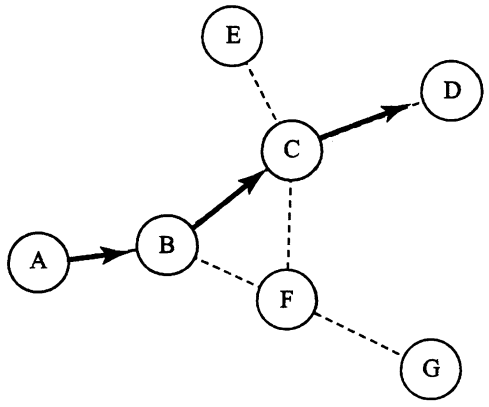


Fig. 3 Singlecast a packet from node A to node D.

2.4 Signal collisions and hidden terminal problem

Signals transmitted by differnt nodes sometimes collides when they are broadcasted or single casted as is described the previous subsection. This collision is classified to two classes: *direct collision* and *indirect collision*. Examples of these collisions are shown in Fig. 4.

Left part of Fig. 4 shows the direct collision between nodes which are connected by communication link (i.e., these nodes are adjacent to each other). In this case, if one node (node A) transmits signal during the signal transmission of the other node (node B). Node B is in transmission process and it cannot receive signal simultaneously (because the communication link uses only one channel, i.e., *half duplex*). Therefore, the signal transmitted from node A cannot be received correctly and is discarded. Similarly, the signal transmitted by node B is discarded at node A.

Right part of Fig. 4 shows the indirect collision occurs among three nodes C, D, and E. Here, node C and D are adjacent to each other, and node D and E are adjacent to each other. However, node C and E are not. In this case, indirect collision occurs when node E transmits signal to node D during the transmission of node C to node D. Signals transmitted from node C and from node E arrive at node D simultaneously. Therefore, both signals collide each other at node D and are discarded by D. In this case, node C and D cannot recognize this collision (because both nodes are without communication ranges of each other) and expect the signals are received by node D correctly. This is the difference of indirect collision from direct one where nodes transmittd signals recognize the signal collision.

To avoid signal collision, CSMA (Carrier Sense Muntiple Access) is commonly used. In this method, each node confirms that there is no signal in wireless channel (i.e., no signal is received) when the node intend to transmit signal, and it transmits based on this cofirmation. On the other hand, if some signal exists in the channel, the signal intended to be transmitted is stored in buffer and is transmitted after some predetermined time passed (this time is called *collision wait time*).

On the other hand, in the case of indirect collision, it is impossible to avoid the collision by CSMA or such a method. Indirect collision is one of essensial problems and called *hidden terminal problem*. This problem may be compensated by re-transmitting the discarded information following the protocol using 'ack' such like TCP in higher layer.

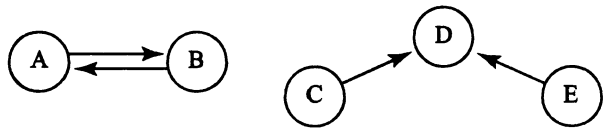


Fig. 4 Direct collision and Indirect collision.

2.5 Re-construction of network

Network topology may be modified along with the movement of nodes. The network should be re-constructed when this modification occurs. This network re-construction includes the following cases:

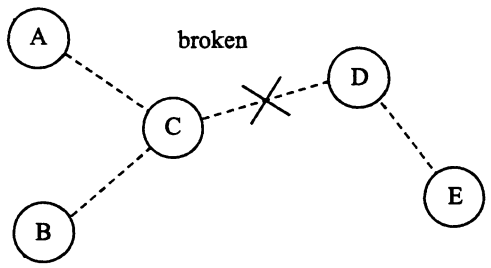
1. The case when a new node joins the network.
2. The case when a node disappeared because of the movement of the node or of some trouble in communication equipment (trasmitter and receiver).
3. The case when a node changes its relative position to others.

In the first case above mentioned, the re-construction of network is completed when the basic entry of new node is broadcasted all over the network (F.T. of the new node itself is completed by updates from adjacent nodes one by one). In spite of the case when some updates disappeares during the broadcasting suffered by the hidden problem, re-construction will be completed after enough time passed because the update is broadcasted repeatedly.

In the second case, there are two processes for re-construct the network. In the first process, each node waits when the 'ist' of the entry becomes older than the predefined value and then the entry is erased. The basic entry of node which is disappeared form the network will not be broadcasted any more. Therefore, 'ist' of every entry which addresses the disappeared node will increase monotonecally and exdeeds critical value (this value is called *entry erase limit*). Finally, all of the entries addresses the disappeared node is erased from the network and then the re-construction is completed.

In the second process, the break of communication link caused by the node disappearance is recognized immediately by another node of communication link. This recognize is done depending on the time interval during which the update from another node is not received (this time interval is called *link break limit*). The node which recognizes this break of the link determines that the other node disappears and broadcasts this information with the update.

An examples are shown in Fig. 5 that of occurring of communication link breakage and that of update broadcasted by the node which recognizes the breakage. This figure indicates the case of breakage of communication link between node C and D in the network with five nodes A $\sim$ E. In this case, node C senses the entry which is addressed to node D becomes older than the link break limit (assumed to be five in this case) because of the long time interval during which update from D is not received. According to this fact, node C determines that the communication link with node D is broken and add all of the entries with 'nxt' equals D to F.T. as the link breakage information. This process contains that 'seq' of each entry is made odd number by added one and that set 'flg' to '*' representing the entry is added to update at the next broadcast.



F. T. of node C

dst	nxt	mtr	seq	ist	flg
C	C	0	22	0	*
A	A	1	20	0	–
B	B	1	22	0	–
D	D	1	13	−5	*
E	D	2	11	−5	*

Fig. 5 Forwarding Table of node C when it senses the breakage of link between node D.

This information of communication link breakage is erased from F.T. just after the next broadcast. The other node which receives this update including the information of communication link breakage decides whether or not the entry is the information of breakage depending on the 'seq' (even or odd number). If the entry is decided as the information of communication link breakage, the node erases the entry form its F.T. just after the information is transmitted as an update. This process is done in the network repeatedly, and after enough time passed, the network re-construction is completed deleting the node D and E.

In the third case, the network re-construction is completed by the combination of processes above mentioned. At the node, which does not move or move but not so long distance as the relative position to other nodes is not change, the entry which addresses the node which changes its relative position to others should be overwrited by the new update. If the information of communication link breakage is received earlier from some adjacent node, once the entry is erased and added by succesive update (transmitted from the new position of the node). On the other hand, if this update is received earlier than the information of communication link breakage, the entry addressed to node which changes its position is overwrited by the update and seccessive information of breakage is ignored (because of the oldness of its 'seq'). One of the two cases above mentioned may occur depending on each node. However, after enough time passed, the re-construction of network is completed anyway.

3. DSDV simulator

In this Section, firstly the overview of the Ad-Hoc network simulator (from now on, referred to as *DSDV simulator*) is explained. Secondly, how to install it, how to compile it, and how to execute it are explained.

3.1 Overview of the simulator

The purpose of DSDV simulator is to simulate the process of construction the network by nodes broadcasting each othe the updates (the transmission of updates are assumed to be unsynchronous among nodes). Therefore, the signals which each node transmits are updates only, and single casted signal addressed to some unique node is not simulated in this version of simulator (the simulation of single casted signals will be included in the functions of the simulator in the next version).

DSDV simulator consists of main function, other functions called by the main one, include files, and initialization file which is loaded at the beginning of execution. Among these contents, the construction of main function is shown by flowchart in Fig. 6 (following descriptions regard the number noted in the figure).

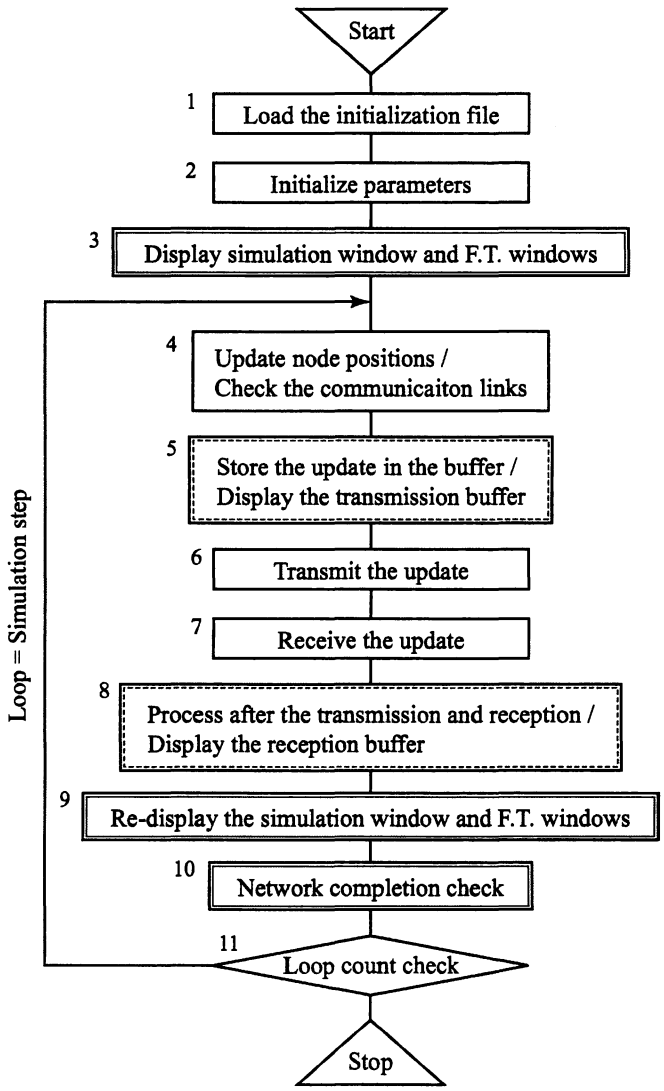


Fig. 6

Flow chart of
DSDV simulator
main function.
(included in dsdv/dsdv.c)

With graphic display
on window

With text display
on termial

1. Fistly, initialization file is loaded and parameters descrived in the file are set to specified initial values.
2. The other parameters are set to the default each.
3. Display the simulation window (simply called *sim. window*), and the same number of F.T. windows as that of nodes. sim. window indicates the states of network and nodes during the simulation. F.T. windows indicate the entries recorded in the F.T. of each node.
4. On the preparation above described, the process enters the simulation loop. In this simulation loop, firstly the positions of each node is updated, and secondly the communication link is reset according to the relative positions of nodess (every processes described in the following is done at each node).
5. If the time for broadcast comes, the temporary update information is stored in the transmission buffer. Also, the contents of transmission buffer is indicated by characters on the terminal emulator in whith the command for the execution of simulator is entered.
6. If the transmission buffer is not empty and if it is not the waiting interval for collision avoiding, broadcast the update. If the node recognizes the other signal transmitted, the node determines the waiting interval and ceases the transmission.
7. Obsserve the signal receiving. And if signal collision is sensed (receives two or more number of signals simultaneously), discards the received signals. On the other hand, no collision occurring is sensed, the received signal is stored in reception buffer.
8. If the update is transmitted, contents of the transmission buffer is shifted and prepare the next transmission. If reception buffer stores the update which is received correctly, the F.T. is updated by the update information. 'ist' of all the entries except for basic entry are examined and generate the communication link breakage information according to the link breakage limit. Erase the entries the oldness of which exceed the entry erase limit.
9. After the processes above described are finished, the states of network and nodes are indicated on sim. window. The contents recorded in F.T. of each node are indicated on F.T. windows.
10. Examine whether or not the network construction is completed, and if it is completed, sim. window indicates the result. Here, the completion of network construction is verified whether or not F.T. of each node is completed by entries of all other nodes in the network (and there is no error in every communication pass).
11. Examine the iteration count and if it exceeds the predefined value, indicate the prompt and wait for the input of next iteration count. Here, the indication style of simulation, that of F.T. and communication pass, and update of moving vector of each node can be specified tempolaly. Or, if necessary, the simulation can be terminated with log files about observed data during the simulation dumped as text files.

Fig. 7 shows the process of transmission of updates from each node unsynchronously during the simulation loop. The figure indicates how the transmission and reception of each node related with the simulation loop when the node A, B, and C construct the linear network as shown in the upper part of the figure. The horizontal axis represents the simulation step indicating the progress of simulation loop (variable 'loop' is used for this step in the simulator).

Firstly, node A transmits update at loop=1, and this transmission keeps five steps and finishes at loop=5. Node B intends to transmit at loop=4. However, because of the reception of signal form node A, it ceases the transmission, set the collision wait time to 3, and transmits later at loop=7. Because the transmission from node A finishes correctly at loop=5, node B updates its F.T. according to the information sent from A. The transmission of node B finishes at loop=11, then node A and C which received this signal from B correctly update their F.T.s. Next, node A and C transmits their updates from loop=12 $\sim$ 18 and loop=13 $\sim$ 17, respectively. However, these signals collide at node B which receives both of the signals (because of hidden terminal problem) and are discarded by node B.

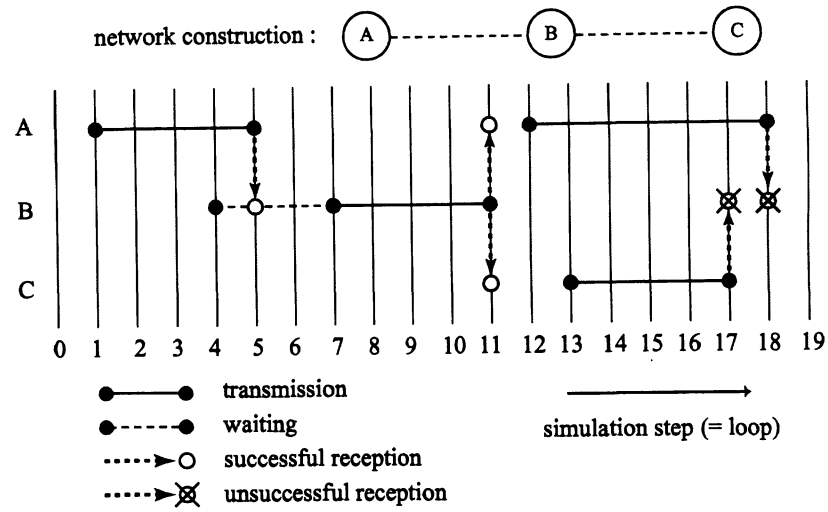


Fig. 7 Simulation step and transmissions.

At the second transmission of node A, number of entries increases because of the update of F.T. jsut before the transmission. As the result, the signal length becomes longer than the first transmission. In DSDV simulator, the signal length is assumed to be propotional to the number of entries its update includes. The constant of the propotionality (viriable 'bcr' is used in the simulator) is specified in initialization file.

Because the time axis is quantized by simulation step, the difference between times of transmission by different nodes cannot be alternated continuously. However, if the constant of proportionality 'bcr' is made large enough and then make the number of simulation step be increased, the process of update transmissions by each node unsynchronously possibly be simulated approximately.

3.2 Installation and execution

DSDV simulator can be downloaded from the following web page.

http://www.comm.info.eng.osaka-cu.ac.jp/ sugi/download/

One 'dsdv.tar.gz' is downloaded from this page, it must be uncompressed and compiled in UNIX workstation. The process of uncompress is:

> gunzip dsdv.tar.gz

> tar xvf dsdv.tar

As the result, directory 'dsdv' is made in current directory (from now on, 'dsdv' is assumed to be the current directory). Then, file: 'Makefile' must be revised. When the file downloaded is uncompressed, thie Makefile is written for Linux. For compiling on the other environment, definitions of macro 'CC' and 'XL' must be changed. For example, if the compiling is done on Solaris, 'X11R6' in both macro must be changed to 'openwin'.

If Makefile is revised correctly, compiling is executed. When the current directory is on some other remote host (for example, the session is doing after remote login some other host), the next command:

> setenv DISPLAY *localhost*:0

must be entered for specifying the X server of local host. Without this command be entered, the compiling will stop when utility 'util/serv' intend to get the information about X server which is not connected with the local host (no need to say that the command: > xhost *remotehost* must be entered on the local host before this procedure and then the access from remote host to the X server is allowed).

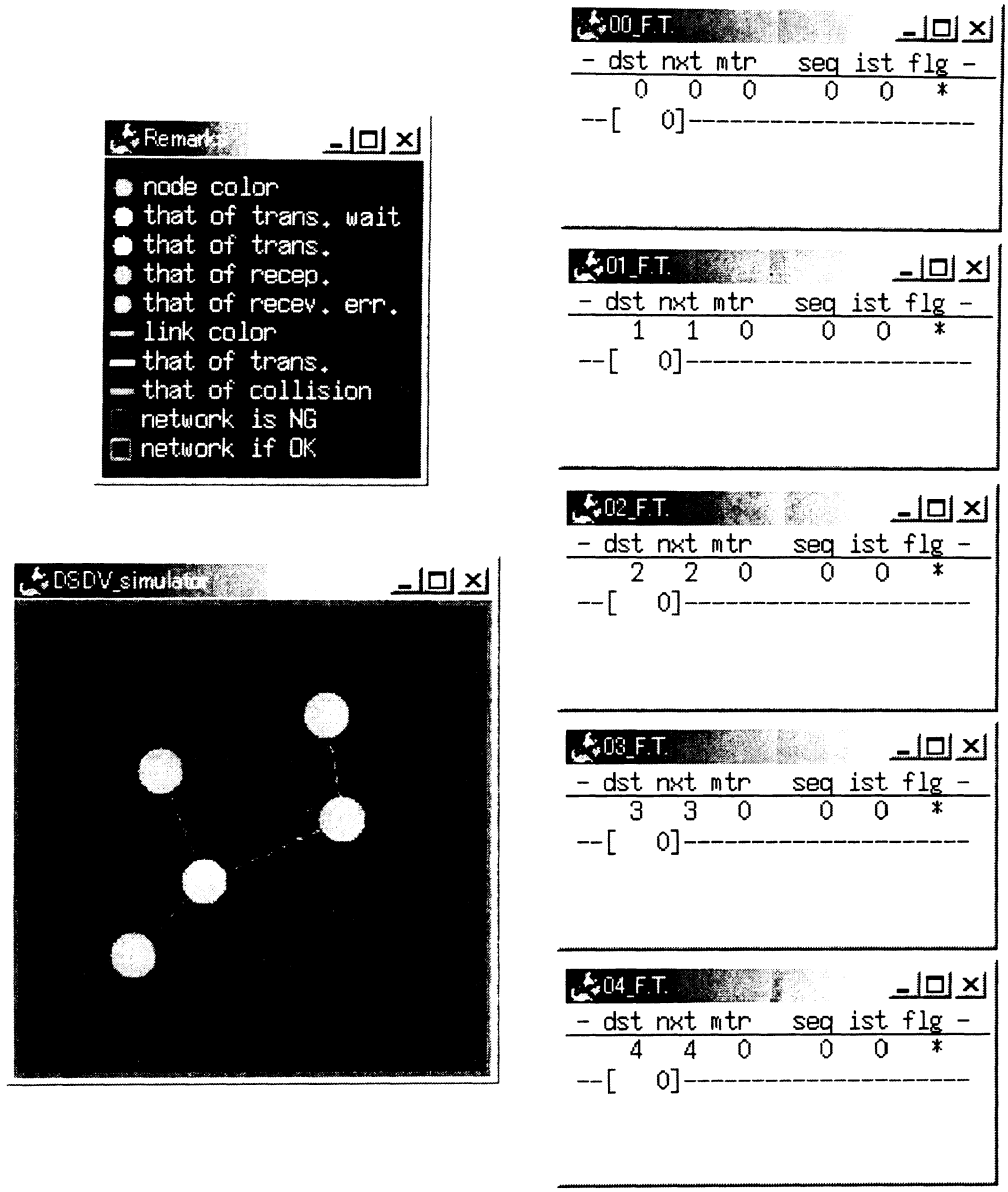


Fig. 8 Windows generated by DSDV-simulator.

```
% ----[DSDV simulator initialization file]--------

% Window width, height, radius of node display [wdt, hit, nodr].
  60  60  3
% Node number and communication radius [N, rng].
   5  20
% Average signal length and its standard deviation [lgm, lgv].
   8   2
% Average period of broadcast and its standard deviation [pdm, pdv].
  50  10
% Coefficient of packet length transmitted from a node.
% packet length = bcr * number of entries in one broadcast [bcr].
   3
% Average wait time for re-transmission when recieved signal
% is sensed, and its standard deviation [wtm, wtv].
  20  5
% Standard for elimination of entry the install time of which:
% node[i].F.T.[j].ist becomes old (entry erase limit).
% And standard for announce as the communication link breakage
% and then erase the entry, when the install time of the entry
% which addressed to adjacent node becomes old (link breakage limit)
% [old, old2].
  500 5
% Seed for random number for the phisical matters such as the initial
% positions of nodes, seed for that for communication processes
% [seed1, seed2]. If '-1' then the seeds are set randomly.
  1234  5678  % 2147483647  2147483647 <- Maximum number
% Color set
----( Omitted )------------------
% Width and height of Remarks window [rmk_w, rmk_h].
  40.0   40.0
% Widht and lines of F.T. window [ftw_w, ftw_n].
  53.0   10
% 1: update the simulation window; 0: does not [simupd].
  1
% 1: display the transmission radius; 0: does not [trdsp].
  0
% 1: display the window of F.T.; 0: does not [ftwdsp].
  1
% 1: transmission buffer and reception buffer are indicated by text.
% 0: are not [bufdsp, rbfdsp].
  1   1
% End of data.
```

On the preparation above, the next command:

> make

execute the compiling and execution file 'dsdv' is made.

To execute the DSDV simulator, only:

> dsdv

is necessary. Then the initialization file 'dsdv.dat' is loaded, and according to the specification written in the file, the simulator is executed. At the beginning of the simulation, three windows named:

1. DSDV_simulator
2. Remarks
3. *nn*_F.T.

appear (examples of these windows are shown in Fig. 8).

Among these, 'DSDV_simulator' is the window which indicates temporary positions of each node and its state in the simulation. Node number is indicated on each node. The state means whether or not each node is transmitting or receiving, and if receiving, whether or not the signal collision occurrs. Moreover, whether or not F.T. of each node is completed by entries adddressed to all other nodes in the network. These states are indicated by display color and border color of window temporary.

'Remarks' shows the relation between the display colors avobe mentioned and what each color represents.

'*nn*_F.T.' indicates the contents of F.T. of each node temporary. Actually, *nn* is replaced by the node number which the window is related with. The number of this windows is the same as that of nodes specified in the initialization file.

The geometries of each window can be specified in initialization file except for the position in the display screen. If windows appear overlapping each other, these must be moved to appropriate positions by mouse operation. At this time, there is the case that the display of windows disappears if the X server does not have the *backing store function*. In this case, 'd' must be entered at the prompt on the terminal emulater on which DSDV simulator is executed. Then, each display will be rewrited.

3.3 Initialization file

Parameters relating to the execution of DSDV simulator are included in initialization file. In this file, parameters are written in predefined order. blank and carriage return separate the parameters. Arbitral number of blanks and carriage returns can be inserted between the neighbouring parameters.

Integer and real number of parameters must be written following the common format for indication. For example, integer number (number of nodes or so) written as 5.0 causes error message and breakage of simulation execution. On the other hand, real number written as 5 or 5.0 is both read correctly. The format for display color is percented values of RGB proportionals with brackets. For example, (100, 100, 100) represents white and (100, 100, 0) represents yellow.

An example of initialization file is shown in the previous page. The line start with % is recognized as a comment. If % exists in a line, the following part of the same line is ignored as a comment. In the initialization file, each comment explain what the parameter means. Each word enclosed by [ ] indicates the variable which is used as each parameter in the simulator. Variables 'lgm' (average signal length) and 'lgv' (its standard deviation) are prepared for the simulation of single cast communication and are not used in the current version of the simulater.

3.4 State indication during the simulation

DSDV simulator indicates the states of each node and network during the simulation on the two windows: DSDV_simulator and *nn*_F.T., on terminal emulater on which the simulater is executed, and on the specified

text files. The appearnce of these indication is shown in the following. Here, the term "input" means the strings put for the prompt:

(xxx) enter (or type 'h' for help):

when the simulation puases (this pause occurs when the iteration count in the simulation exceeds the specified value). Where, 'xxx' is replaces by the temporal count of iteration.

(1) indication of state of each node

F.T. (window) Displays temporal F.T. of node *nn* on *nn*_F.T.. (terminal emulator) If 'f' is entered, then temporal F.T. of each node is indicated succesively. (file) If 'q' is entered, then simulation is terminated and F.T. of each node is dumped into the file 'FwdTbl.log'.

transmission buffer (terminal emulator) If return key is put simply, then transmission buffer of each node is indicated succesively at each simulation step. If 'b-' is entered, then this indication is restricted.

reception buffer (terminal emulator) If return key is put simply, then reception buffer of each node is indicated succesively at each simulation step. If 'r-' is entered, then this indication is restricted.

(2) indication of state of network

communication pass (terminal emulator) If 's' is put, then temporal communication passes start form each node are indicated succesively. Errors or breakages in the communication pass is indicated simultaneously. (file) If 'q' is put, then simulation is terminated and temporal communication passes start form each node are dumped to file 'RoutChk.log'.

network completion (window) If F.T. of each node is completed with entries addressed to all other nodes in the network, border color of the window: DSDV_simulator is changed. (terminal emulator) If 's' is put, then the first line indicated at the time shows whether or not the network is completed and the value of 'loop' provided the network is completed. (file) If 'q' is put, then simulation is terminated and file: 'RoutChk.log' is generated. In the first line of this file, it is shown whether or not the network is completed and the value of 'loop' provided the network is completed.

state of communication (window) The state of each node and that of communication link in the network are indicated on the window: DSDV_simulator. (terminal emulator) If 'c' is put, then the state of communication of each node: transmission time, transmission step, time for transmission waiting, step count of transmission waiting for collision avoidance, reception time, step of the reception, collision time when receiving, and step count during the collision is occurring accumulated so far are indicated succesively. (file) If 'q' is put, then simulation is terminated and file: 'ColWit.log' which records above accumulated state of node and network is made.

Fig. 9 indicates above mentioned.

3.5 Proceeding of the simulation

The prompt mentioned before is indicated at the beginning of simulation. If some integer value is put here, then simulation progresses for the specified number of steps. If other character is put, then the process is done depending on the character, and states of nodes and network, state of simulation, and such information are indicated. If 'h' is put, then the relation of the character put for the prompt and the process incured by the character is shown as follows.

	Display of node state			Display of network state		
	F.T.	Transmisson buffer	Receive buffer	Routes	Network construction	Transmission state
Window	Real time on *nn*_F.T. windows.	——	——	——	DSDV_simulator window as the frame of it.	When type 'c'.
Terminal	When type 'f'.	Every simulation step.	Every simulation step.	When type 's'.	When type 's'.	When type 'c'.
File	"FwdTbl.log" when type 'q'.	——	——	"RoutChk.log" when type 'q'.	The first line of "RoutChk.log".	"ColWit.log" when type 'q'.

Fig. 9 Display of node state and network state during simulation by Windows, Terminals, and Files.

```
-------------------------------------------------
[RET]: step is not changed (initial value: 1).
  100: step is changed to 100.
   s : display the state of network.
   b : from now on, the state of buffer is shown.
   b-: from now on, that is not shown.
   r : from now on, that of receive buffer is shown.
   r-: from now on, that is not shown.
   x : from now on, X-graphics is updated.
   x-: from now on, that is not updated.
   f : forwarding table is indicated.
   c : collisions and wait times are reported.
 m 3 0.2 60: moving step of node 3 is changed
             to 0.2mm/step to 60deg. direction.
   d : re-display windows.
   i : initialize parameters.
   q : quit.
   h : help.
-------------------------------------------------
```

For example, if simulation is desired to proceed rapidly, enters of 'b-' and 'r-' (divided into tow times) restrict the indication of transmission buffer and reception buffer at each simulation stp and then cause the desired rapid progress of the simulation. Moreover, enter of 'x-' restrict the graphics operation of X server. If the indication of buffers and graphics operation is disired to be done again, put 'b', 'r', and 'x' for the next prompt indicated.

If 'm' is put as the first character, the succesive values set the moving vector of specified node. For example,

> m 3 0.2 60

causes the vector of length 0.2[mm] and direction 60[degree] set for the node 3. This function is used to simulate the network topology modification occurres when some node moves and changes its relative position to other nodes in the network. If node 3 is desired to be stable again,

> m 3 0 0

causes the desired state of node 3.

4. Conclusions

Ad-Hoc network simulator is introduced for the graphical simulation on the display screen of UNIX workstation. Three types of windows: simulation window, F.T. window and Remarks window are generated which indicate the state of the node and the network, entries recorded in forwarding table of each node, and some remarks for identifying the display colors on the simulation window, respectively. Simulation data are dumped to text files including transmissin collision time and wait time for avoiding that at each node when the simulation is terminated.

In the next version, simulation of single cast packet transmission addressed to unique node will be included in the functions of this simulator. This function may include the investigation of transmission characterisitcs when packets are transmitted and verified following TCP protocol. In this case, signal collisions between data packets and acknowledge packets must be occur frequently, and as the result, transmission characteristics may be degraded seriously[4]. This problem should be examined by DSDV simulator in next version.

References

1) J. P. Hubaux, T. Gross, J. L. Boudec, and M. Vetterli, "Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project," *IEEE Commun. Mag.*, vol. 39, no. 1, pp.118–124 (2001)

2) C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pp. 234-244 (1994)

3) C. E. Perkins and E. M. Royer, "Ad Hoc on-demand distance vector routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100 (1999)

4) M. Gerla, K. Tang, R. Bagrodia. "TCP Performance in Wireless Multihop Networks," *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pp.41–50 (1999)